Chino Valley Unified School District
High School Course Description

A. CONTACTS	
1. School/District Information:	School/District: Chino Valley Unified School District Street Address: 5130 Riverside Dr., Chino, CA 91710 Phone: (909) 628-1201 Web Site: chino.k12.ca.us
2. Course Contact:	Teacher Contact: Anthony Pittman Position/Title: Teacher Site: Ayala High School Phone: (909) 627-3584 E-mail: anthony_ pittman@chino.k12.ca.us

B. COVER PAGE - COURSE ID	
1. Course Title:	Advanced Placement (AP) Computer Science Applications (CSA)
2. Transcript Title/Abbreviation:	AP CSA
3. Transcript Course Code/Number:	5E40
4. Seeking Honors Distinction:	AP Distinction
5. Subject Area/Category:	Meets "g" a-g UC/CSU elective requirement
6. Grade Level(s):	11-12
7. Unit Value:	10 credits/ 5 credits per semester
8. Course Previously Approved by UC:	Yes
9. Course Classified as a Career Technical Education Course:	No
10. Course Modeled after a UC- Approved Course:	Yes
11. Repeatable for Credit:	Yes
12. Date of Board Approval:	March 2, 2017

13. Brief Course Description:

AP Computer Science Applications (AP CSA) aligns with the College Board's Computer Science Applications framework and the PLTW (Project Lead the Way) Computer Science framework. AP CSA builds on the basic skills learned in AP Computer Science Principles (AP CSP) to teach students Java and authentic Android app development. Students in this course continue to hone their communication and collaboration skills while learning to use a variety of tools. The primary goal of the course is to create independent-thinking app developers: every unit in this course builds on students' prior knowledge and skills until they can complete an app development cycle independently from the ground up.

AP CSA is designed to cover all learning objectives in the College Board's AP Computer Science Applications framework, and exceeds the College Board's requirement of 20 hours of lab activity. It is also an example of a CSTA (Computer Science Teachers Association) level 3C course.

The course is designed to be readily adaptable to student interests and community assets. Individual teachers are encouraged to modify the course content so that it feels as authentic and meaningful within the local context as possible. This course aims to fully develop Object Oriented Programming (OOP) skills that were introduced in AP Computer Science Programming.

14. Prerequisites:	Integrated Math 3/3H or completion of AP CSP

15. Context for Course:
This class is designed to further student ability in programming. Instruction includes the learning and practice several programming languages. It also develops awareness and skill in app development for mobile devices. This class will provide the hands-on training and practice in these skills. Students will be working on apps they develop in a project-based-learning model of instruction.

16. History of Course Development:
AP Computer Science Applications had been around for more than 10 years as the AP CSAB (Computer Science Accreditation Board) exam, which was discontinued in 2008 and replaced with the CSA exam. The exam and materials are constantly evolving. The most recent iteration of the exam uses the Google Android platform to develop full-fledged apps using the coding language Java.

17. Textbooks:	None
18. Supplemental Instructional Materials:	Access to computers with appropriate software and access to mobile devices

C. COURSE CONTENT

1. Course Purpose:
AP Computer Science Applications emphasizes object-oriented programming methodology with an emphasis on problem solving and algorithm development and is meant to be the equivalent of a first-semester course in computer science. It also includes the study of data structures and abstraction.

1. Object-Oriented Program Design:
The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, and can be adapted to changing circumstances. The design process needs to be based on a thorough understanding of the problem to be solved.
a. Program and Class Design

2. Program Implementation:
Part of the problem-solving process is the statement of solutions in a precise form that invites review and analysis. The implementation of solutions in the Java programming language reinforces concepts, allows potential solutions to be tested, and encourages discussion of solutions and alternatives.
a. Implementation techniques
b. Programming constructs
c. Java library classes and interfaces included in the AP Java subset

3. Program Analysis:
The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms to understand their time and space requirements when applied to different data sets.
a. Testing
b. Debugging
c. Runtime exceptions
d. Program correctness
e. Algorithm Analysis
f. Numerical representations of integers

4. Standard Data Structures:
Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures.
a. Primitive data types (int, Boolean, double)
b. Strings
c. Classes
d. Lists
e. Arrays (1-dimensional and 2-dimensional)

5. Standard Algorithms:
Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency.
a. Operations on data structures
b. Searching
c. Sorting

6. Computing in Context:
An awareness of the ethical and social implications of computing systems is necessary for the study of computer science. These topics need not be covered in detail, but should be considered throughout the course.
a. System reliability
b. Privacy
c. Legal issues and intellectual property
d. Social and ethical ramifications of computer use

For more detail on the course topics covered in Computer Science A, see the Course Description:
https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-a-course-description.pdf

2. Course Outline:
Unit 1: Introducing Java:
Unit 1 provides a primer in the basics of the Java programming language and Object Oriented Programming (OOP). Students create classes, instantiate them, add instance data and access that data. They use conditionals, iteration, arithmetic and logical operators, arrays and iterators, first in BlueJ to ensure code correctness, and then in Android Studio to incorporate their own code into fully functional apps. The material provided also includes extra practice on all these Java topics and more.
Introducing Java

Lesson 1.1 Objects in Java:
The goal of this lesson is to give students the tools they need to create Java objects. Students create their own methods and call them to manage and manipulate data. Students then program the logic into a weather app that notifies the user of appropriate courses of action to take when heading out the door in the morning based on the weather forecast. The lesson concludes with students augmenting the artificially intelligent natural language processing Magpie app based on the College Board’s Magpie Chatbot lab.
a. Activity 1.1.1 – Introduction to Android Development
b. Activity 1.1.2 – Your First Class
c. Activity 1.1.3 – Making Objects
d. Activity 1.1.4 – If It’s Raining...
e. Activity 1.1.5 – Your Sci-Fi Name
f. Activity 1.1.6 – Chatting with Magpie

Lesson 1.2 Manipulating Data:
This lesson focuses on managing data in arrays and using iteration in Java. Students write code in BlueJ that parses string data and then plug that code into an app in Android Studio that retrieves data from the device's local memory. They also write code to manage and maintain a list of billboard music ratings. The unit culminates with students creating an app from the ground up that uses buttons to play sound assets.

a. Activity 1.2.1 – Parsing Text
b. Activity 1.2.2 – Today's Top 40
c. Activity 1.2.3 – Data Storage
d. Activity 1.2.4 – Create an Android Project
e. Activity 1.2.5 – Synthesizer

Unit 2: Vanilla Android Development:
Students spend most of this unit developing a viewer for college applications, with college admissions officials as the target audience. Highlights of this unit include working with fragments, mastering encapsulation, and designing and implementing apps that incorporate the most common and useful user interface elements. Students use a Backend as A Service (BaaS) to implement persistent data within their app, allowing a user to access their data from any Android device. At the end of the unit, students design apps and perform usability testing on their designs using a prototyping tool.

Lesson 2.1 App Navigation:
Students begin this lesson by testing out a sample app that showcases the functionality of the final product they are asked to produce in the Unit 2. The College App is designed to quickly show an admissions officer whatever assets an applicant has provided in a mobile format. Students learn to incorporate and extend several common User Interface (UI) features into the College App. In the process, they learn about inheritance and class definitions and have an opportunity to apply prior knowledge of basic Java constructs. Incorporating a navigation drawer into the app improves usability and gives students experience working with a design pattern found in many real-world apps. Finally, students explore and critique a Unified Modeling Language (UML) diagram and other documentation for the College App.

a. Activity 2.1.1 – Usability Testing
b. Activity 2.1.2 – Prototyping with proto.io
c. Activity 2.1.3 – Classes
d. Project 2.1.4 – App Navigation
e. Activity 2.1.5 – User Input

Lesson 2.2 Data Persistence:
This lesson continues to emphasize the OOP paradigm, reinforcing previous learning. Additionally, students learn about and use some common data structures including Array Lists. Students create their own checked exceptions, and access a Backend as a Service (BaaS) to implement data persistence. They create classes that inherit from interfaces or other classes and use these within data structures, necessitating a solid understanding of polymorphism. Finally, students decide on a feature to add to the College App and implement it.

a. Activity 2.2.1 – Exceptions and Scope
b. Activity 2.2.2 – Remote Database
c. Activity 2.2.3 – List View
d. Problem 2.2.4 – One Method, Many Classes
e. Activity 2.2.5 – List and Detail
f. Project 2.2.6 – Integration Testing and Unit Testing (4 days)

Lesson 2.3 The Development Process:
Students start out this lesson by accessing Google's libraries for taking and displaying pictures in the College App. Having completed the College App, students will have fully utilized the Array List class in one dimension, extended interfaces and abstract classes, overloaded methods and more. This lesson culminates with students choosing a project in which they will extend their knowledge. They will revisit this project at the end of the course when they know more about Android development. Students must prototype and test their app for usability, as well as properly document and present their final products.

a. Activity 2.3.1 – Let Me Take a Selfie
b. Activity 2.3.2 – The Development Process

Unit 3: Advanced Android Features:
The goal of Unit 3 is for students to reach a level of understanding of Google's Android libraries that allows them to create apps using a broad range of mobile features such as Global Positioning Systems (GPS) and Internet services. The major project in this unit is a social networking app that utilizes the BaaS they learned in Unit 2. Students begin by learning to manage a new set of data in the back-end database, writing client and server code in their app. Students then learn to access the GPS features of mobile devices, to reading QR codes (Quick Response codes) and accessing the web. The unit culminates in a problem in which students create a geo-cache style app using the techniques they dev

Lesson 3.1 Trip Tracker:
In this lesson, students learn to add a backend service to their apps. Students store and retrieve user data from the cloud. This provides teachers with an opportunity to connect the OOP that students have learned in Java to authentic web frameworks and APIs (Application Programming Interfaces). The goal of this lesson is to give students the power to create apps that store significant amounts of data, that benefit from crowd/cloud-sourcing, and then allow users to access data from anywhere. Students manage the front and back end interfaces for a social networking app, going through the steps of prototyping and usability testing. They continue to enhance the social networking app throughout this unit.

a. Activity 3.1.1 – Trip Tracker Start Up
b. Activity 3.1.2 – User Authentication
c. Activity 3.1.3 – A New Trip
d. Activity 3.1.4 – Listing Trips
e. Activity 3.1.5 – Updates and Deletes
f. Activity 3.1.6 – Public vs. Private Trips
g. Activity 3.1.7 – Sort Algorithms
h. Activity 3.1.8 – Search Algorithms
i. Project 3.1.9 – Social Networking App – Design
j. Project 3.1.10 – Social Networking App – Development

Lesson 3.2 Location Awareness:
In this lesson students reinforce their understanding of basic Java language constructs and OOP while making their apps location aware with GPS. They also practice data storage and management by adding location data to posts that users make in the app.

a. Activity 3.2.1 – Preparing for Google Play Services
b. Activity 3.2.2 – Using Google Play Services
c. Project 3.2.3 – Location Awareness App – Design
d. Activity 3.2.4 – Location Awareness App – Development (4 days)

Lesson 3.3 Contacts in an App:
This lesson concludes work on the Trip Tracker app by adding the ability to query the device to discover locally stored contacts. Each contact is categorized into a subclass of an abstract type and then combined into a large list, requiring students to use and understand polymorphism. To achieve the finished product, students must use common features such as auto boxing, common methods from the String class, and dynamic late binding, in addition to reinforcing all the Java constructs from the previous unit. Students create a GeoQuest app that combines the camera feature from Unit 2, the geolocation feature from Lesson 3.2 and knowledge of polymorphism, lists, and strings. The app keeps track of a list of polymorphic quest items to find around campus; when a team member discovers a quest item, the user records it in the app with a geolocation tag and an image of the item. When all required items have been found, the quest is complete.

a. Activity 3.3.1 – Trip Cost and Rating
b. Activity 3.3.2 – Polymorphic People
c. Activity 3.3.3 – Persistent People
d. Activity 3.3.4 – Polymorphic Behavior
e. Activity 3.3.5 – Geo-Quest

Lesson 3.4 App Analysis:
The goal of this lesson is to explore nuances of the Java language, especially those that occur during computational algorithms. Students will analyze the performance of various sorts and searches, perform statement execution counts, learn a simple rounding algorithm, experiment with operator precedence, witness integer overflow, and convert between the hexadecimal and decimal number systems.

a. Activity 3.4.1 – Investigating Sort
b. Activity 3.4.2 – Investigating Search
c. Activity 3.4.3 – Computations in Java

Unit 4: The LibGDX Game Development Framework:
The goal of Unit 4 is to give students an opportunity to practice and refine their understanding of Java techniques in the context of game development. LibGDX is a popular open source game development framework that is constantly growing due to the contributions of its active community members. An important part of this lesson is teaching students to access resources to help themselves utilize all the tools that are available to them. Students learn to incorporate media assets, and work with graphics and touch events. Students access and manipulate data in 2D data structures, and interpret code created using the MVC pattern before making significant modifications of their own. Finally, students create a unique app that incorporates elements like geolocation, communication with a database, and utilization of the camera, speakers, and microphone. The choice of app theme and topic are left to the student, though they should target a specific audience, and benefit their community in some way.
Students will need to find a "client" with whom they will communicate regularly about the progress of the project. This could be the manager of a GitHub repository, a community leader, or a local business owner. Students might also choose to develop a full-fledged game at this point in the course with their fellow students as the clients.

Lesson 4.1 Creating a New World:
The goal of this lesson is to get students to understand the foundations of game development in LibGDX. The product uses the touch screen to register user input. Students incorporate 2D graphic assets into the project, and manipulate 2D data structures.

a. Activity 4.1.1 – LibGDX Setup
b. Activity 4.1.2 – Level Loading
c. Activity 4.1.3 – Walls, Characters, and Doodads

Lesson 4.2 Graphic Adventure Game:
In this lesson, students transfer their knowledge to fix problems with existing source code and add entirely new features to an existing game. To meaningfully improve the existing code, students must use math and problem solving skills as well as Java and object oriented concepts covered previously in this course.

a. Activity 4.2.1 – Code Overview
b. Activity 4.2.2 – Erratic Movement
c. Project 4.2.3 – Game Improvement

Lesson 4.3 Independent Projects:
Before the start of this lesson, students may opt to investigate bonus activities that cover animation outside of LibGDX, and publish apps they develop. In this lesson students put into practice everything they've explored in the course thus far. Likely more than for any task they've taken on up to this point, this problem requires students to be effective teammates, collaborators, communicators, and developers. Students will be tasked with finding an authentic "client" for their work with whom regular communication is embedded in the Agile design process. This project may be a continuation of student work from Unit 2. Student projects should address authentic needs in their community, though the choice of problem is left to the individual students. Example projects might include a water conservation awareness app for clients living in the drought-riddled southwest, or a puzzle game for friends to play. Students might choose to make an app that allows students to register online for courses at their school, develop an educational game for younger students, or perhaps even build an interactive set of tutorials for this very course. The possibilities are limitless, but the time available is not. Students will need to practice expert time management skills to create successful apps.

a. Problem 4.3.1 – Make an App

Android is a trademark of Google Inc.

3. Key Assignments:
One 2.5-3 week project each semester

4. Instructional Methods and/or Strategies:
Project Lead the Way APB (Activity, Project, and Problem-based) Instructional Design providing students with unique opportunities to work collaboratively, identify problems, apply what they know, persevere through challenges, find unique solutions, and lead their own learning.

5. Assessment Including Methods and/or Tools:

- AP Computer Science Applications Course Description (PDF) (Opens in new window)
- AP Computer Science Applications quick reference (PDF) (Opens in new window)

The exam is three hours long and has two sections — multiple choice and free-response.
Students will not be tested on minor points of syntax. All code given is consistent with the AP Java subset. All responses involving code must be answered in Java. The exam also includes a quick reference to both the multiple-choice and free-response sections of the exam.

Section I: Multiple Choice | 40 Questions | 1 hour and 30 minutes | 50% of Final Exam Score
Question topics will include:

- Programming Fundamentals
- Data Structures
- Logic
- Algorithms/Problem Solving
- Object-Oriented Programming
- Recursion
- Software engineering

Section II: Free-Response | 4 Questions | 1 hour and 30 minutes | 50% of Final Exam Score

Short answer questions, each requiring Java programming language.